Documentation supplement 2.3

CHAPTER 1

INTRODUCTION

This chapter introduces PRO/Tool Kit BASIC-PLUS-2 and shows how it fits into the Professional family of software. You should read this chapter first; then, based on what you read here and on your current level of knowledge about the topics discussed, read elsewhere in the Professional documentation before you attempt to use PRO/Tool Kit BASIC-PLUS-2.

This chapter discusses the following topics:

- Section 1.1 introduces PRO/Tool Kit BASIC-PLUS-2 and its components.
- Section 1.2 previews the installation and verification procedures for PRO/Tool Kit BASIC-PLUS-2.
- Section 1.3 lists the steps that you must follow to develop an application on the Professional.

1.1 PRO/TOOL KIT BASIC-PLUS-2

PRO/Tool Kit BASIC-PLUS-2 is a modified version of the BASIC-PLUS-2 compiler and Object Time System (OTS). The BASIC-PLUS-2 resident library, BP2V23.TSK, provides run-time support for applications.

The function and use of PRO/Tool Kit BASIC-PLUS-2 and BASIC-PLUS-2 on RSX are very similar; however, some important differences exist. These differences are discussed in detail in Chapter 4.

In general, once you have installed the Professional Developer's Tool Kit and PRO/Tool Kit BASIC-PLUS-2, you can use your Professional to develop BASIC-PLUS-2 applications that run on P/OS.

The PRO/Tool Kit BASIC-PLUS-2 package includes the following components:

- The PRO/Tool Kit BASIC-PLUS-2 Compiler
- The PRO/Tool Kit BASIC-PLUS-2 Object Time System (OTS), which includes:
  - The PRO/Tool Kit BASIC-PLUS-2 Resident Libraries, BP2V23.TSK and BP2V23.STB
  - The PRO/Tool Kit BASIC-PLUS-2 Disk Library, BP2OTS.OLB
- The PRO/Tool Kit BASIC-PLUS-2 Help Files
- The PRO/Tool Kit BASIC-PLUS-2 Resequencer

- The PRO/Tool Kit BASIC-PLUS-2 Error Message Files
- The PRO/Tool Kit BASIC-PLUS-2 Overlay Descriptor Files (ODL)
- The documentation
  - PRO/Tool Kit BASIC-PLUS-2 Installation Guide and Documentation Supplement
  - BASIC on RSX-11/M-PLUS Systems
  - BASIC Reference Manual
  - BASIC User's Guide
  - BASIC Pocket Reference Guide

1.2 INSTALLATION AND VERIFICATION

The PRO/Tool Kit BASIC-PLUS-2 compiler is distributed in the form of a prebuilt task image. This image is linked against the resident libraries, B23SHR.TSK and B23SH1.TSK. You cannot edit a command build file to tailor PRO/Tool Kit BASIC-PLUS-2 to your environment. Similarly, the Object Time System task is prebuilt and cannot be modified. Installation simply consists of copying files from the distribution volume to appropriate directories.

If you want to modify the compiler to suit your system environment, you can preset compiler defaults using the compiler initialization file. Section 4.3 describes the initialization file in detail.

Using files supplied during the installation of PRO/Tool Kit BASIC-PLUS-2, the Installation Verification Procedure (IVP) guides you through the creation of a sample PRO/Tool Kit BASIC-PLUS-2 application that verifies the installation.

1.3 SUMMARY OF THE APPLICATION DEVELOPMENT PROCESS

The development of an application requires a number of steps, some of which may have to be repeated in a cycle. The Tool Kit User's Guide describes all these steps.

Here is a list of steps you must follow; more detailed information is presented in Section 4.2. Additionally, the PRO/Tool Kit BASIC-PLUS-2 verification application, which is described in Chapter 3, guides you through the following steps:

1. Writing the source program
2. Compiling the source program
3. Generating the command and overlay descriptor files
4. Building the P/OS task image
5. Running and debugging the task image
6. Writing the application installation file
7. Installing, running, and debugging the application
8. Creating the final diskette

CHAPTER 4

DEVELOPING THE APPLICATION

This chapter supplements the BASIC on RSX-11M/M-PLUS Systems manual and is intended for programmers experienced with BASIC-PLUS-2 on RSX systems. For complete information about developing BASIC applications, see the other manuals included in the PRO/Tool Kit BASIC-PLUS-2 documentation set. The sections in this chapter provide the following information:

- Section 4.1 summarizes the differences between PRO/Tool Kit BASIC-PLUS-2 and BASIC-PLUS-2 on RSX.
- Section 4.2 explains each of the steps that you need to perform to develop an application using PRO/Tool Kit BASIC-PLUS-2.
- Section 4.3 explains how you can preset compiler defaults for PRO/Tool Kit BASIC-PLUS-2, using a compiler initialization file.
- Section 4.4 describes and lists PRO/Tool Kit BASIC-PLUS-2 run-time error messages.

4.1 DIFFERENCES BETWEEN PRO/TOOL KIT BASIC-PLUS-2 AND BASIC-PLUS-2 ON RSX

For the most part, the function and use of PRO/Tool Kit BASIC-PLUS-2 and BASIC-PLUS-2 on RSX are very similar; however, some important differences exist. These differences are summarized in the next section.

4.1.1 Differences and Unsupported Features

Table 4-1 summarizes items that you must consider when you write PRO/Tool Kit BASIC-PLUS-2 applications. Some of the items in the table are features that operate differently than on RSX systems, others are not supported. Do not use the unsupported features when you write PRO/Tool Kit BASIC-PLUS-2 applications.

Table 4-1: PRO/Tool Kit BASIC-PLUS-2 -- Differences and Unsupported Features

Feature		Please Note
Compiler Commands		
	RUN	This command is not available in the Tool Kit compiler.
	LOAD	This command is not available in the Tool Kit compiler.
	BUILD	There are differences between the CMD and ODL files created by PRO/Tool Kit BASIC-PLUS-2 and PDP-11 BASIC-PLUS-2 on RSX systems.
	BUILD/DUMP	This command is unsupported.
	BUILD/NOSEQUENTIAL	This command is unsupported for programs that do not access RMS files.
	SET DUMP	This command is unsupported.
	SET NOSEQUENTIAL	This command is unsupported for programs that do not access RMS files.
Debugger Commands		
	REDIRECT	This command allows debugging during application execution from a terminal attached to the Professional. It does not accept a terminal device number.
Error Messages[1]		The Tool Kit compiler error messages are a subset of BASIC-PLUS-2 on RSX error messages. You must press RESUME to continue after receiving an error message. Note that error number 135 is issued if you attempt indexed I/O on a P/OS Diskette application (as opposed to a P/OS Hard Disk application).
Immediate Mode		Immediate mode is not available in the Tool Kit compiler.
$ Command		Supported for RSX-11M/M-PLUS systems only (not VMS).
Magnetic Tape Operations		Magnetic tape operations are unsupported.

1. For a complete list of PRO/Tool Kit BASIC-PLUS-2 error messages, see Section 4.4.

(Continued on next page)

Table 4-1 (Cont.): PRO/Tool Kit BASIC-PLUS-2 -- Differences and Unsupported Features

Feature	Please Note
RMS Support	Magnetic tape operations are unsupported.
Statements	
CHAIN	CHAIN accepts as an argument the installed task name, which is the same name used in the application installation file. CHAIN does not accept the file name of the task image.
EDIT$ Function	You cannot use an odd integer as the argument to EDIT$ if the string to be operated on contains an 8-bit character. If you do, the high-order bit for that character is cleared.
TIME$ Format	The twelve-hour AM/PM format for the TIME$ function is unsupported.
CTRL/C Trapping	Enabling CTRL/C trapping causes the terminal to be attached. Some of the callable P/OS routines cannot be called if the terminal is attached (see the Tool Kit User's Guide).
DECnet File Access (DAP)	This feature is unsupported.

4.1.2 Character Sets

PRO/Tool Kit BASIC-PLUS-2 supports the 8-bit DEC Multinational Character Set in string literals, comments, and DATA statements when you compile an application, and as input or output when you run the application on the Professional. (See the Terminal Subsystem Manual for a description of the DEC Multinational Character Set.)

The PRO/Tool Kit BASIC-PLUS-2 compiler does not allow European alphabetics in variable names.

If the string to be operated on contains an 8-bit character, do not use any odd integer as the argument to the EDIT$ function. If you do, the high-order bit for that character is cleared.

4.2 DEVELOPMENT CYCLE

This section explains each of the steps you need to perform to develop an application using PRO/Tool Kit BASIC-PLUS-2. Note, however, that the descriptions in this chapter are restricted to aspects of the development process that are unique to PRO/Tool Kit BASIC-PLUS-2.

The development cycle for PRO/Tool Kit BASIC-PLUS-2 applications (Figure 4-1) consists of the following steps:

1. Writing the source program
2. Compiling the source program to create an object module file:
   - Compiling with /DEBUG to debug at run time
   - Compiling without /DEBUG when the program is error free
3. Generating the command and overlay descriptor files
4. Building the P/OS task image
5. Running and debugging the task image file
6. Writing the application installation file
7. Installing, running, and debugging the application
8. Running the Application Diskette Builder, when the program is error free, to create the final diskette

The following sections present specific details of application development with PRO/Tool Kit BASIC-PLUS-2. For a complete description of each stage in Tool Kit application development, refer to the Tool Kit User's Guide. Also note that, with the Tool Kit, you can create application diskettes that can run on P/OS Diskette, on P/OS Hard Disk, or on both. This document focuses on application development for the P/OS Hard Disk environment. Some important differences in the application development cycle for these two environments exist. Refer to the Tool Kit User's Guide for full details.

4.2.1 Writing the Source Program

You can create a BASIC source program by entering text in the BASIC environment or by using any text editor available on the Professional. Refer to the appropriate text editor manual for more information.

Your PRO/Tool Kit BASIC-PLUS-2 source program can include external subroutine calls to access P/OS facilities from your program. The external subroutine calls are described in the Tool Kit User's Guide and the CORE Graphics Library Manual. P/OS facilities exist in the following areas:

1. P/OS user interface support -- Your application has access to P/OS menu, help, message, and other system services. The ability to display error messages, status, online help, and menus on the screen is provided through the P/OS Service Routines Library (POSRES).
2. Extended software functions -- Your application can invoke callable system services, such as PROSE (the P/OS text editor) and PRO/Communications (the Professional communications facility).
3. Bitmap graphics routines -- Your application has access to the graphics routines in the CORE Graphics Library.

P/OS facilities are accessed through the register 5 (R5) calling sequence with the PRO/Tool Kit BASIC-PLUS-2 CALL BY REF statement.

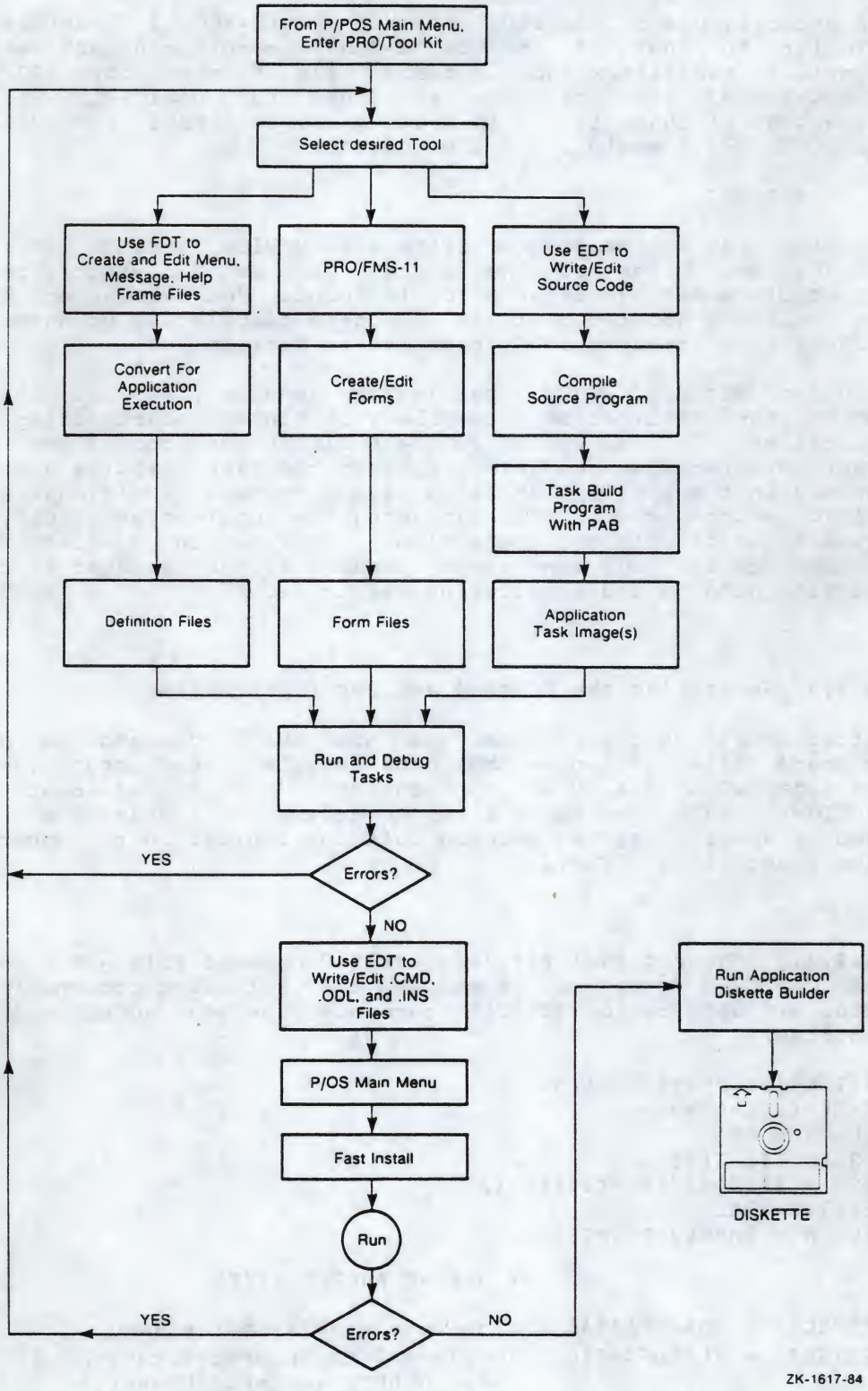

Figure 4-1: PRO/Tool Kit BASIC-PLUS-2 Development Cycle

4.2.2 Compiling the Source Program

In general, use of the PRO/Tool Kit BASIC-PLUS-2 compiler is very similar to that of the BASIC-PLUS-2 compiler on RSX systems. The compiler qualifiers and command line formats for PRO/Tool Kit BASIC-PLUS-2 are the same as those for BASIC-PLUS-2, with the exception of those listed in Table 4-1. To invoke the PRO/Tool Kit BASIC-PLUS-2 compiler, use the command:

```
$ BASIC
```

Compile your source program using the COMPILE command. The compiler checks each line of the source program for errors, returning an appropriate message if an error is found. You can correct the program as necessary and recompile it. Program compilation produces an object module file (filename.OBJ) that can be task built.

PRO/Tool Kit BASIC-PLUS-2 has its own debugger. To use the debugger with your application, compile your source program using the /DEBUG qualifier. To find errors in the BASIC source code, you can debug your program when you have completed the task building step. To find errors in the application files, and perhaps additional errors in BASIC source code, you can debug the application itself, after you have built it with the Professional Application Builder (PAB). In either case, you can issue commands to the debugger to control and monitor program and application execution.

4.2.3 Generating the Command and Descriptor Files

After compiling the program, use the BUILD command to generate a command file (filename.CMD) and overlay descriptor language file (filename.ODL) for your application. The Professional Application Builder (PAB) uses these files to define how libraries are referenced and to specify special-purpose buffers, logical unit numbers (LUNs), and event flags (EFNs).

4.2.3.1 The PRO/Tool Kit BASIC-PLUS-2 Command File - The PRO/Tool Kit BASIC-PLUS-2 compiler generates the following command file for the Tool Kit application "TEST1" (comments have been added to describe the entries):

```
SY:TEST1/CP=SY:TEST1/MP
TASK=task-name
UNITS = 18
ASG = TI:13:15
ASG = SY:5:6:7:8:9:10:11:12
EXTTSK= 952
CLSTR = RMSRES,POSRES:RO

                          ; DEFINE BUFFER SIZES

EXTSCT   = DM$BUF:4540    ; dynamic single-choice menu
EXTSCT   = FL$BUF:4310    ; file selection/specification
                          ; for OLDFIL and NEWFIL routine
EXTSCT   = HL$BUF:3410    ; help text/menu
EXTSCT   = MM$BUF:1000    ; multi-screen menu
EXTSCT   = MN$BUF:4540    ; static single-choice menu
```

```
                        ; DEFINE LUN ASSIGNMENTS

GBLDEF  = HL$LUN:21     ; help frame file
GBLDEF  = MN$LUN:20     ; menu frame file
GBLDEF  = MS$LUN:16     ; message frame file
GBLDEF  = TT$EFN:1      ; terminal I/O event flag
GBLDEF  = TT$LUN:15     ; terminal I/O
GBLDEF  = WC$LUN:22     ; directory searches for OLDFIL and NEWFIL
                        ; routine or callable print services
//
```

Each line in the file specifies a command or option to the task builder. You can specify additional options; however, the default options supplied in the BASIC CMD file let you access the callable routines available on the Professional. See the Tool Kit User's Guide and the RSX-11M/M-PLUS Task Builder Manual for more information about CMD files.

4.2.3.2 The PRO/Tool Kit BASIC-PLUS-2 Overlay Descriptor File - The PRO/Tool Kit BASIC-PLUS-2 compiler generates the following overlay descriptor language file for the Tool Kit application "TEST1":

```
            .ROOT       BASIC2-RMSROT-USER,RMSALL
USER:       .FCTR       SY:TEST1-LIBR
LIBR:       .FCTR       LB:[001005]BP2OTS/LB
@LB:[001005]BP2IC1
@LB:[001005]RMSRLX
            .END
```

See the Tool Kit User's Guide and the BASIC on RSX-11M/M-PLUS Systems manual for more information about ODL files.

4.2.4 Building the P/OS Task Image

After you have created the desired CMD and ODL files, use the Professional Application Builder (PAB) to create an application task image (filename.TSK). Invoke PAB and enter the command file as follows:

```
$ RUN $PAB
PAB>@TEST1
```

This creates the task image TEST1.TSK.

See the Tool Kit User's Guide for complete information on Professional Application Builder commands.

4.2.5 Running and Debugging the Task Image File

After you create an executable task image, you can run it with the RUN command and use the PRO/Tool Kit BASIC-PLUS-2 debugger to debug the program. (See your BASIC documentation for more information on the debugger.)

When you execute a task that has been compiled with the /DEBUG qualifier, you can debug it by redirecting the debugger output to a terminal connected to the printer port on the Professional. (See the Tool Kit User's Guide for details.) Use the PRO/Tool Kit BASIC-PLUS-2

REDIRECT command to display debugger I/O and issue debugger commands on that terminal. Input is accepted from and output goes to the printer port. Note that the REDIRECT command does not accept a terminal device number. All other I/O, including graphics or forms, remains unaffected.

4.2.6 Writing the Application Installation File

You must write an installation file (filename.INS) before you can install the application on the Professional. The installation file for the application "TEST1" looks like this:

```
! PRO Installation file TEST1.INS
NAME 'Test1 Application'
FILE [TEST1]TEST1.TSK/DELETE
INSTALL [TEST1]TEST1.TSK/TASK
RUN TEST1
```

Some P/OS services require additional lines in the INS file. See the Tool Kit User's Guide for information specific to the service you plan to use and for a description of what should go into the INS file. In particular, note the differences between applications running under P/OS Diskette and those running under P/OS Hard Disk.

4.2.7 Installing, Running, and Debugging the Application

Use the P/OS menu structure and Fast Install to install your new application. When you execute Fast Install, you are prompted to enter a directory name. Enter the name of the directory that contains the INS and TSK files (and other required files); note that the name of the INS file must be identical to the directory name. Press DO when you have entered the directory name.

Fast Install now prompts you to select a menu group within which to install the new application. After you have chosen the menu group, you have an opportunity to change both the menu application name and the group name. Press DO when you are satisfied with both names.

Fast Install reports the success or failure of the installation process. Refer to the Tool Kit User's Guide for more information.

When the application has been successfully installed, you can run it using the P/OS menu structure. If the BASIC object module was compiled with the /DEBUG qualifier, you can also debug the application by using the REDIRECT command and a terminal attached to the Professional (see Section 4.2.5).

4.2.8 Running the Application Diskette Builder (ADB)

When your application is error free, run the Application Diskette Builder to create a final diskette. See the Tool Kit User's Guide for complete information on Application Diskette Builder commands.

4.3 PRESETTING THE COMPILER ENVIRONMENT WITH AN INITIALIZATION FILE

To preset the compiler environment, you can use a compiler initialization file. The compiler initialization file (called the BP2INI file) allows you to override the defaults chosen when PRO/Tool Kit BASIC-PLUS-2 was installed. Thus, you can preset the compiler environment to suit your needs. BP2INI files contain compiler commands that set defaults for libraries, for COMPILE and BUILD commands, for ODL files, and so on. Section 4.3.2 lists the compiler commands that are valid in a BP2INI file.

When you invoke BASIC-PLUS-2, the compiler searches for BP2INI files in two locations. First, the compiler searches for a file named BP2INI.BP2 in LB:[001002]. If it finds this file, the compiler executes the commands in it. Then, the compiler searches for a file named BP2INI.BP2 in the current default directory. If it finds this file, the compiler executes the commands in it. Note that if the same type of command is executed in both files, the command in your current default directory BP2INI file overrides the command executed in the LB:[001002]BP2INI.BP2 file.

This scheme allows you great flexibility; you can change defaults for every invocation of BASIC-PLUS-2 by modifying the BP2INI file in LB:[001002]. And you can set up specialized defaults by keeping additional BP2INI files in other directories.

The next three sections are organized as follows:

- Section 4.3.1 describes how to create a BP2INI file.
- Section 4.3.2 describes the commands allowed in a BP2INI file.
- Section 4.3.3 is a sample BP2INI file.

4.3.1 Creating a BP2INI File

Use a text editor to create a BP2INI file for PRO/Tool Kit BASIC-PLUS-2, and name the file BP2INI.BP2. You can include the commands listed in Section 4.3.2. Each command must appear on a separate line. Note that comments are not allowed in the BP2INI file; however, you can include blank lines to improve readability.

To set up the system compiler initialization file, that is, a BP2INI file that takes effect each time you invoke PRO/Tool Kit BASIC-PLUS-2, you must create a file named BP2INI.BP2 in the LB:[001002] directory.

Since the compiler searches for a second BP2INI file in the current directory, you can create additional BP2INI files in any number of directories. The BP2INI file in your directories can contain commands that override the commands in the system initialization file (BP2INI.BP2 in LB:[001002]) and the commands that make additional changes to the compiler environment.

Note that both the system and current directory initialization files are optional. The PRO/Tool Kit BASIC-PLUS-2 installation does not produce an initialization file; without one, the compiler environment is determined by the installation defaults.

You can display the installation defaults by entering the SHOW command from within the PRO/Tool Kit BASIC-PLUS-2 environment:

```
$ BASIC

PDP-11 BASIC-PLUS-2 V2.3-00

BASIC2

SHOW
PDP-11 BASIC-PLUS-2 V2.3-00 using FPU with NO run support
ENVIRONMENT INFORMATION:          RMS FILE ORGANIZATION:
    Current edit line : 0             NO Index
    NO Modules loaded                 NO Relative
    NO Main module loaded                Sequential
                                      NO Virtual
DEFAULT DATA TYPE INFORMATION:    LISTING FILE INFORMATION:
    Data type : REAL                  NO Source
    Real size : SINGLE                NO Cross Reference
    Integer size : WORD               NO Keywords
    Scale factor : 0                     60 lines by 132 columns
COMPILATION QUALIFIERS:           BUILD QUALIFIERS:
       Object                         NO Dump
    NO Macro                          NO Map
       Lines                             Cluster
       Warnings                       NO I- and D-Space
    NO Debug records                  Task extend  : 512
    NO Syntax checking                RMS ODL file : LB:[1,5]RMSRLX
    Flag : Declining                  BP2 Disk lib : LB:[1,5]BP2OTS
    Variant : 0                       BP2 Resident lib : NONE
                                      RMS Resident lib : RMSRES

BASIC2
```

4.3.2 BP2INI File Commands

The following commands are allowed in a PRO/Tool Kit BASIC-PLUS-2 compiler initialization file. They are a subset of PRO/Tool Kit BASIC-PLUS-2 compiler commands:

Command	Function
LOCK	Identical to the SET command
SCALE	Sets the scale factor for COMPILE or BUILD
SET	Sets compiler defaults
SHOW	Displays current defaults
BRLRES	Sets BASIC memory-resident library

For a complete description of these commands, see your BASIC documentation.

No other commands are allowed. If a BP2INI file includes an illegal command, an error message is generated when the compiler is invoked. For example, if you try to use the invalid command NEW, you get this message:

```
? NEW is an Illegal command from the initialization file.
```

4.3.3 Sample BP2INI.BP2 File

The format of the BP2INI file is a simple list of compiler commands that begin in the first column. For example:

```
BRLRES BP2V23
SET /LIST
SET /DOUBLE/BYTE
SCALE 4
SHOW
```

This BP2INI.BP2 file presets the compiler environment so that:

- A source listing is produced (SET /LIST).
- The default size for all floating-point data is DOUBLE, and the default size for all integers is BYTE (SET /DOUBLE/BYTE).
- The scale factor is set to 4 (SCALE 4).
- The compiler defaults are displayed when you enter the compiler environment (SHOW command).

If you create this sample file as the system compiler initialization file (LB:[001002]BP2INI.BP2), and no BP2INI file exists for your current default directory, the following text is displayed whenever you invoke the compiler:

```
$ BASIC
PDP-11 BASIC-PLUS-2 V2.3-00 using FPU with NO run support
ENVIRONMENT INFORMATION:          RMS FILE ORGANIZATION:
    Current edit line : 0             NO Index
    NO Modules loaded                 NO Relative
    NO Main module loaded                Sequential
                                      NO Virtual
DEFAULT DATA TYPE INFORMATION:    LISTING FILE INFORMATION:
    Data type : REAL                     Source
    Real size : DOUBLE                NO Cross Reference
    Integer size : BYTE               NO Keywords
    Scale factor : 4                  60 lines by 132 columns
COMPILATION QUALIFIERS:           BUILD QUALIFIERS:
       Object                         NO Dump
    NO Macro                          NO Map
       Lines                             Cluster
    NO Debug records                  NO I- and D-space
    NO Syntax checking                Task extend  : 512
    Flag : Declining                  RMS ODL file : LB:[1,5]RMSRLX
    Variant : 0                       BP2 Disk lib : LB:[1,5]BP2OTS
                                      BP2 Resident lib : NONE
                                      RMS Resident lib : RMSRES
```

Note the changes from the installation default environment that was previously displayed. The defaults for "Real size :", "Integer size :", "Scale factor :", and " Source" are changed by the initialization file.

Compiler commands within a BP2INI file function the same as those in the compiler environment, with two exceptions:

- BASIC appends the text "in initialization file" to regular error messages to notify you that the error occurred in an initialization file. For example:

```
?Illegal library name specified in initialization file
```

- Commands that normally prompt for an argument do not do so; instead, the following error is reported:

```
no file specified for command in initialization file
```

The contents of BP2INI file are not automatically displayed at the terminal. If you want to display the contents of the file, you should use the SHOW command as the last command in your file. Thus, each time you invoke the compiler, the default settings are displayed. If you do not include the SHOW command as the last command in the file, the defaults shown do not reflect all the changes produced by the file.

For complete information on the BP2INI file, see the BASIC on RSX11-M/M-PLUS Systems manual.

4.4 PRO/TOOL KIT BASIC-PLUS-2 RUN-TIME ERROR MESSAGES

PRO/Tool Kit BASIC-PLUS-2 run-time error messages are a subset of BASIC-PLUS-2 run-time error messages on RSX systems. The PRO/Tool Kit BASIC-PLUS-2 run-time error messages are preceded by a number. If you use the ERT$ function to reference an error number that is not in the Tool Kit BASIC-PLUS-2 subset, ERT$ returns the following message:

```
Can't access LB:[001002]BP2ERR.ERR or can't find frame <id>
```

Frame <id> identifies the error number of the ERT$ you tried to reference.

If the debugger is not present, that is, if the program was not compiled with the /DEBUG qualifier, the following notice appears on the screen after the error message is displayed:

```
Please write down the above message -- press RESUME to continue.
```

You must press the RESUME key to clear the screen and continue.

Table 4-2: PRO/Tool Kit BASIC-PLUS-2 Error Messages

Number	Text
1	?Bad directory for device
2	?Illegal file name
3	?Account or device in use
4	?No room for user on device
5	?Can't find file or account
6	?Not a valid device
7	?I/O channel already open
8	?Device not available
9	?I/O channel not open
10	?Protection violation
11	?End of file on device
12	?Fatal system I/O failure
13	?User data error on device
14	?Device hung or write locked
15	?Keyboard wait exhausted
19	?Disk block is interlocked

(Continued on next page)

Table 4-2 (Cont.): PRO/Tool Kit BASIC-PLUS-2 Error Messages

Number	Text
28	?Programmable ^C trap
31	?Illegal byte count for I/O
34	?Reserved instruction trap
35	?Memory management violation
43	?Virtual array not on disk
44	?Matrix or array too big
45	?Virtual array not yet open
46	?Illegal I/O Channel
47	?Line too long
48	%Floating point error
50	%Data format error
51	%Integer error
52	?Illegal number
53	%Illegal argument in LOG
54	%Imaginary square roots
55	?Subscript out of range
56	?Can't invert matrix
57	?Out of data
58	?ON statement out of range
59	?Not enough data in record
60	?Integer overflow, FOR loop
61	%Division by 0
63	?FIELD overflows buffer
64	?Not a random access device
72	?RETURN without GOSUB
73	?FNEND without function call
88	?Arguments don't match
89	?Too many arguments
97	?Too few arguments
104	?RESUME and no error
105	?Redimensioned array
116	?PRINT USING format error
126	?Maximum memory exceeded
127	%SCALE factor interlock
130	?Key not changeable
131	?No current record
132	?Record has been deleted
133	?Illegal usage for device
134	?Duplicate key detected
135	?Illegal usage
136	?Illegal or illogical access
137	?Illegal key attributes
138	?File is locked
139	?Invalid file options
140	?Index not initialized
141	?Illegal operation
142	?Illegal record on file
143	?Bad record identifier
144	?Invalid key of reference
145	?Key size too large
147	?RECORD number exceeds maximum
148	?Bad RECORDSIZE value on OPEN
149	?Not at end of file
150	?No primary key specified
151	?Key field beyond end of record
152	?Illogical record accessing

(Continued on next page)

Table 4-2 (Cont.): PRO/Tool Kit BASIC-PLUS-2 Error Messages

Number	Text
153	?Record already exists
154	?Record/bucket locked
155	?Record not found
156	?Size of record invalid
157	?Record on file too big
158	?Primary key out of sequence
159	?Key larger than record
160	?File attributes not matched
161	?Move overflows buffer
162	?Cannot open file
164	?Terminal format file required
166	?Negative fill or string length
167	?Illegal record format
168	?Illegal ALLOW clause
170	?Index not fully optimized
171	?RRV not fully updated
173	?Invalid RFA field
180	?No support for op in task
183	?REMAP overflows buffer
184	%Unaligned REMAP variable
185	?RECORDSIZE overflows MAP buffer
186	?Improper error handling
227	?String too long
246	?Error trap needs RESUME
247	?Illegal RESUME to subroutine
248	?Illegal return from subroutine
250	?Not implemented
251	?Recursive subroutine call
252	?FILE ACP failure
253	?Directive error